factorial

Script:

```
#!/bin/bash

fact() {
n=$1
if [ "$n" -le 1 ]; then
echo 1
return
fi
res=1
for ((i=2;i<=n;i++)); do
res=$(( res * i ))
done
eccho "$res"
}

if [ $# -lt 1 ]; then
echo "Usage: $0 <non-negative integer>"
exit 1
fi

for arg in "$@"; do
if ! [[ $arg =~ ^[0-9]+$ ]]; then
echo "$arg: not a non-negative integer, skipping."
continue
fi

echo "$arg! = $(fact "$arg")"
done
```

Breakdown:

1. `#!/bin/bash`--> Shebang: tells the system to run the script with Bash.
2. `fact() { … }`--> Defines a function named fact that calculates a factorial.
   - `n=$1`-> Stores the first argument to the function in variable n.
   - `if [ "$n" -le 1 ]; then echo 1` -> if n is less than 1, print 1.
   - `return`-> to exit the function.
   - `res=1`-> Initializing result variable.

- `for ((i=2;i<=n;i++)); do res=$(( res * i )) done echo "$res"`-> Start of a for loop from 2 to n. tthen multiplying res by i on each iteration and then ending the loop.
- `echo "$res"`-> Output the computed factorial.

3. `if [ $# -lt 1 ]; then echo "Usage: $0 <non-negative-integer>`-> Check if the script got fewer than 1 argument and showing usage if no arguments.
4. `exit 1`-> tells the shell to stop running the current script. The number (1) is the exit code or return status.
5. `for arg in "$@"; do`-> Loop through all command-line arguments.
6. `if ! [[ $arg =~ ^[0-9]+$ ]]; then`-> To check if arg is mon negative integer.
7. `continue`-> Skip to next argument.
8. `echo "$arg! = $(fact "$arg")"`-> Call fact and print the factorial.
9. `done`-> End the main loop.

Output:

Case 1: Single number:

```
./factorial.sh 5
5! = 120
```

Case 2: Several numbers:

```
./factorial.sh 0 1 3 6
0! = 1
1! = 1
3! = 6
6! = 720
```

Case 3: Valid and invalid inputs:

```
./factorial.sh 4 hello -2
4! = 24
hello: not a non-negative integer, skipping.
-2: not a non-negative integer, skipping.
```